creating-a-custom-theme

Custom Theme How to Create a Custom Theme: A Step-by-Step Guide ■ Introduction ■ This guide shows you how to apply and dynamically switch between brand themes in a Play+ application. Built on the Metamorphic pillar and powered by a token-driven architecture, Play+ supports pure CSS-based theming — with automatic derivation of hover, focus, dark mode, and accessibility tokens. All users need to do is modify a single file: _custom-theme.css . No SCSS. No boilerplate. Overview ■ Play+ ships with a default color palette, fonts, spacing, and radii — known as global tokens . You don't need to define these unless you're adding custom values. To theme your app: Pick from the provided tokens in _custom-theme.css (Optional) Add custom raw values directly in the CSS Switch themes at runtime with a single line of code Folder Structure Example ■ src/ ■■■ styles/ ■■■ tokens/ ■■■ themes/ ■■■ _default.css ■■■ _enterprise.css ■■■ _acme.css ■■■ _custom-theme.css Step 1: Define Your Brand Theme via CSS ■ Create a _custom-theme.css file for your brand. Each semantic token is mapped either to: A Play+ pre-defined global token name (e.g., var(--global-color-blue-500)), or A raw value (e.g., #007BFF , "Inter, sans-serif" , "8px") /* styles/tokens/themes/_custom-theme.css */ [data-theme = "custom"] { /* PRIMARY (Your Brand Color) */ --color-brand-primary : var (--global-color-blue-500) ; --color-brand-primary-hover : var (--global-color-blue-700) ; --color-surface-interactive-primary : var (--global-color-blue-500) ; --color-border-primary : var (--global-color-blue-500) ; /* SECONDARY (Accent Color) */ --color-brand-secondary : var (--global-color-teal-600) ; --color-brand-secondary-hover : var (--global-color-teal-700) ; --color-surface-interactive-secondary : var (--global-color-teal-100) ; --color-border-secondary : var (--global-color-teal-600) ; /* BACKGROUND & SURFACE */ --color-background-primary : var (--global-color-white) ; --color-background-secondary : var (--global-color-gray-50) ; --color-surface-interactive-default : var (--global-color-blue-500) ; --color-surface-disabled : var (--global-color-gray-200) ; /* BORDERS */ --color-border-default : var (--global-color-gray-300) ; --color-border-interactive : var (--global-color-blue-500) ; --color-border-focus : var (--global-color-blue-500) ; --color-border-error : var (--global-color-red-500) ; } That's all you need to do to brand your app.

Play+ will handle hover, contrast text, focus rings, and dark mode derivations automatically. Step 2: Load the Theme Dynamically ■ In your app's entry point (e.g., main.ts , index.tsx , or equivalent), load your desired theme by setting the data attribute: // main.ts or index.tsx document . documentElement . setAttribute ("data-theme" , "custom") ; This tells Play+ to apply the mappings defined in styles/tokens/themes/_custom-theme.css . You can switch themes based on tenant ID, user preferences, or URL parameters. This will apply styles/tokens/themes/_custom-theme.css Switching themes per tenant, user, or route becomes trivial Step 3: Default Theme Fallback ■ If no theme is selected, Play+ automatically falls back to the default theme: styles/tokens/themes/_default.css What Play+ Does Behind the Scenes ■ Once --color-brand-primary is set, Play+ will automatically derive: --color-brand-primary-hover : (darkened or lightened) --color-text-on-brand-primary : (accessible contrast color) --color-border-focus : based on context dark mode variants : using the same token map with transformations This reduces errors, increases consistency, and eliminates boilerplate. Best Practices ■ Do: ■ Only edit _custom-theme.css Use named semantic roles like --color-brand-primary , --global-radius-md , --font-family-heading Keep themes self-contained Use predictable folder structure (themes/_custom-theme.css) Don't: ■ Modify core Play+ theme internals Write hardcoded values in your components Duplicate logic for hover, focus, etc. Try to derive variants yourself — let Play+ handle that With this CSS-first approach, Play+ makes it possible to theme entire apps through configuration — with runtime switching, zero SCSS, and built-in intelligence for accessibility and state management. Just change the CSS. Play+ takes care of the rest.